In the Specification:

Please amend the specification as follows:

Please amend paragraph 2 under the heading, "TECHNICAL FIELD," as follows:

[2] Embodiments of the ~~The~~ present invention relate ~~relates~~ to the field of digital signal processing. More particularly, embodiments of the invention relate ~~relates~~ to a device and method for providing an FFT/IFFT implementation providing minimum inter-processor communication overhead and less silicon area in a multiprocessor system.

Please amend paragraph 4 as follows:

[4] The DFT of a sequence of length N can be decomposed into successively smaller DFTs. The manner in which this principle is implemented falls into two classes. The first class is called a "decimation in time" approach and the second is called a "decimation in frequency" method. The first derives its name from the fact that in the process of arranging the computation into smaller transformations the sequence "x(n)" (the index 'n' is often associated with time) is decomposed into successively smaller subsequences. In the second general class the sequence of DFT coefficients "x(k)" is decomposed into smaller subsequences (k denoting frequency). Embodiments of the ~~The~~ present invention employ ~~employs~~ "decimation in time".

Please amend paragraph 6 as follows:

[6] The conventional method of implementing an FFT or Inverse Fast Fourier Transform (IFFT) uses a radix-2 / radix-4 / mixed-radix approach with either "decimation in time (DIT)" or a "decimation in frequency (DIF)" approach.

Please amend paragraphs 15 and 16 as follows:

[15] An ~~aspect~~ embodiment of the present invention is to overcome the above drawbacks and provide a device and method for implementing FFT/IFFT with minimum communication overhead among processors in a multiprocessor system using distributed memory.

[16] According to an ~~aspect~~ embodiment of the invention, a scalable method for implementing FFT/IFFT computations in multiprocessor architectures that provides improved throughput by eliminating the need for inter-processor communication after the computation of the first "log2P" stages for an implementation using "P" processing elements, comprises the steps of:

computing each butterfly of the first "log2P" stages on either a single processor or each of the "P" processors simultaneously,

distributing the computation of the butterflies in all the subsequent stages among the "P" processors such that each chain of cascaded butterflies consisting of those butterflies that have inputs and outputs connected together, are processed by the same processor.

Please amend paragraphs 20-23 as follows:

[20] Another ~~aspect~~ embodiment of the present invention provides a system for obtaining scalable implementation of FFT/IFFT computations in multiprocessor architectures that provides improved throughput by eliminating the need for inter-processor communication after the computation of the first "$\log_2 P$" stages for an implementation using "P" processing elements, comprising:

a means for computing each butterfly of the first "$\log_2 P$" stages on either a single processor or each of the "P" processors simultaneously,

an addressing means for distributing the computation of the butterflies in all the subsequent stages among the "P" processors such that each chain of cascaded butterflies consisting of those butterflies that have inputs and outputs connected together, are processed by the same processor.

[21] According to one ~~aspect~~ embodiment of the present invention, the addressing means comprises addresses generation means for deriving the operand addresses of the butterflies subsequent to the first "$\log_2 P$" butterflies in such a manner that the butterfly is processed by the same processor that computed the connected butterfly of the previous stage in the same chain of butterflies.

[22] According to one ~~aspect~~ embodiment of the present invention, the address generation means is a computing mechanism for deriving the address of the first operand in the operand pair corresponding to the "i^{th}" stage of the computation from the address of the corresponding operand in the previous stage by inserting a "0" in the "$(i+1)^{th}$" bit position of the address, and deriving the address of the second operand by inserting a "1" in the "$(i+1)^{th}$" bit position of the operand address.

[23] The above system further includes a computing mechanism for address generation of twiddle factors for each butterfly on the corresponding processor according to a further ~~aspect~~ embodiment of the present invention.

Please amend paragraph 24 as follows:

[24] Embodiments of the ~~The~~ present invention will now be explained with reference to the accompanying drawings, which are given only by way of illustration and are not limiting for the present invention.

Please amend paragraph 30 as follows:

[30] **FIG. 2** shows the implementation for a 16 point FFT in a 2-processor architecture using an embodiment of the present invention. Dark lines are computed in one processor, and light lines in the other. The computational blocks are represented by '0'. The left side of each computational block is its input (the time domain samples) while the right side is its output (transformed samples). Embodiments of the ~~The~~ present invention use ~~uses~~ a mixed radix approach with decimation in time. The first two stages of the radix-2 FFT/IFFT are computed as a single radix-4 stage. As these stages contain only load/stores and add/subtract operations there is no need for

multiplication. This leads to reduced time for FFT/IFFT computation as compared to that with full radix-2 implementation. The next stages have been implemented as radix-2. The three main nested loops of conventional implementations have been fused into a single loop which iterates "$N/2*(\log_2 N-2))/(\text{number of processor})$" times. Each processor is used to compute one butterfly in one loop iteration. Since there is no data dependency between different butterflies in this algorithm, both during and between stages, the computational load can be equally divided among the different processors, leading to a nearly linear scalable system. There is no data dependency between stages and therefore each processor is able to perform the butterfly computations on the data assigned to it without communicating with the other processors.